CLEMENT SETH ROBERTS (STATE BAR NO. 209203)
croberts@orrick.com
BAS DE BLANK (STATE BAR NO. 191487)
basdeblank@orrick.com
ALYSSA CARIDIS (STATE BAR NO. 260103)
acaridis@orrick.com
EVAN D. BREWER (STATE BAR NO. 304411)
ebrewer@orrick.com
ORRICK, HERRINGTON & SUTCLIFFE LLP
The Orrick Building
405 Howard Street
San Francisco, CA 94105-2669
Telephone: +1 415 773 5700
Facsimile: +1 415 773 5759

SEAN M. SULLIVAN (*pro hac* vice)
sullivan@ls3ip.com
MICHAEL P. BOYEA (*pro hac* vice)
boyea@ls3ip.com
COLE B. RICHTER (*pro hac* vice)
richter@ls3ip.com
LEE SULLIVAN SHEA & SMITH LLP
656 W Randolph St., Floor 5W
Chicago, IL 60661
Telephone: +1 312 754 0002
Facsimile: +1 312 754 0003

Attorneys for Sonos, Inc.

UNITED STATES DISTRICT COURT

NORTHERN DISTRICT OF CALIFORNIA

SAN FRANCISCO DIVISION

GOOGLE LLC,

Plaintiff and Counter-defendant,

v.

SONOS, INC.,

Defendant and Counter-claimant.

Case No. 3:20-cv-06754-WHA
Related to Case No. 3:21-cv-07559-WHA

SONOS, INC.'S OPPOSITION TO GOOGLE'S MOTION FOR SUMMARY JUDGMENT PURSUANT TO THE COURT'S PATENT SHOWDOWN PROCEDURE

Date: June 9, 2022
Time: 8:00 a.m.
Place: Courtroom 12, 19th Floor
Judge: Hon. William Alsup

Complaint Filed: September 28, 2020

FILED UNDER SEAL

TABLE OF CONTENTS

Page(s)

TABLE OF AUTHORITIES

Page(s)

Cases

Other Authorities

I. INTRODUCTION

Google's motion for summary judgment should be denied because Google infringes claim 13 of the '615 Patent and claim 1 of the '885 Patent and Google has not met its burden to demonstrate that claim 13 of the '615 Patent is invalid or because, at a minimum, there are material issues of fact in dispute regarding infringement and validity.

Google's summary of the parties' collaboration is irrelevant to summary judgment and inaccurate. Google reached out to Sonos in 2011 inquiring about Sonos's technology. Google reached out to Sonos again in June 2013 inquiring as to Sonos's willingness to collaborate on a set of projects. Prior to collaborating with Sonos, Google had not considered using a cloud to maintain a playback queue and did not have an implementation of such a feature. But this was not a new idea to Sonos. Years before, Sonos had already conceived of using the cloud to maintain a playback queue and had already described the idea in independent patent filings in 2011, 2012, and 2013 (all prior to the collaboration with Google), including the very filing from which the '615 and '033 Patents claim priority. Thus, it is inconceivable for Google to argue that Sonos misappropriated anything from Google in June 2013. To the contrary, while Sonos and Google collaborated, Google secretly worked to undermine this collaboration by incorporating features (including Sonos-patented features) into new products that would compete with the very product on which the parties were collaborating.

Threaded throughout Google's motion is the false premise that a multi-device system can have either a "cloud queue" or a "local playback queue" but not both. This is not true, as demonstrated by Google's own admissions. In truth, Google has invented this false dichotomy to distract from its infringing functionality and push its false misappropriation tale.

II. LEGAL STANDARD

Summary judgment is inappropriate "if the evidence is such that a reasonable jury could return a verdict for the nonmoving party." *Anderson v. Liberty Lobby, Inc.*, 477 U.S. 242, 248 (1986). At summary judgment, "the court does not make credibility determinations or weigh conflicting evidence. Rather, it draws all inferences in the light most favorable to the nonmoving party." *Soremekun v. Thrifty Payless, Inc.*, 509 F.3d 978, 984 (9th Cir. 2007).

III. ARGUMENT

A. Google Infringes Claim 13 of the '615 Patent

Claim 13 of the '615 Patent is directed to a computer-readable storage medium with executable instructions that enable a "control device"[1] to perform recited functions for transferring media playback from the "control device" to a "playback device." Sonos accuses Google of infringement for providing "control devices" (e.g., smart phones, tablets, etc.) provisioned with any of (i) the YouTube, YouTube Music, YouTube Kids, or YouTube TV apps (collectively, "the YouTube apps") or (ii) the GPM app. Google often refers to these devices as "Senders." Ex. 1, ¶¶48-49. Each Sender is provisioned with the accused "Cast" technology that enables it to transfer media playback responsibility from itself to a Cast-enabled media player (i.e., "playback device"), which Google typically refers to as a "Receiver" or "Cast Device." *Id.*

Google argues that it does not infringe '615 claim 13 because its YouTube and GPM apps fail to satisfy limitation 13.5(a), which recites "causing one or more first cloud servers to add multimedia content to a local playback queue on the particular playback device" by adding "one or more resource locators corresponding to respective locations of the multimedia content at one or more second cloud servers of a streaming content service." In short, Google argues that its Receivers do not have a "local playback queue" and do not have a queue with "multimedia content" or "resource locators." As explained below, Google's arguments are without merit, and at the very least, there are genuine issues of fact regarding these arguments that precludes summary judgment.

1. Casting Involves a "Local Playback Queue" on the Receiver

Contrary to its attorneys' arguments here, Google has routinely admitted that a Receiver has a local queue that dictates its playback when in a Cast session. For example, Google's source code includes comments indicating that Google's engineers view a Receiver as maintaining a local queue. *See, e.g.*, Ex. 2, lns. 35-36 ("Notifies the receiver about any local changes to the cloud queue and asks the ***receiver***[2] to refresh ***its queue***."); Ex. 3, lns. 526-29 ("videoQueuer" "[m]anages the ***video queue*** and transitions."); /2021-02-01_YTReceivers09292020/…/application.ts, lns. 3519-22 ("isQueueEmpty" "True if there are no videos in the queue"). Google's engineers also internally

[1] Sonos refers to the claim limitations herein according to Google's labeling.
[2] All emphasis has been added herein unless otherwise noted.

describe a Receiver as having a queue. *See, e.g.*, Ex. 4, 2009 ("[R]eceivers can ***retain a local version of the remote queue***…. It sounds like the queue is intentionally retained to guard against flaky networks ….") Consistent with Google's internal recognition, Google publicly describes a Receiver as maintaining a playback queue. *See, e.g.*, Ex. 5, 66 ("The ***Receiver SDK maintains the queue*** and responds to operations on the queue as long as the queue has at least one item currently active (playing or paused)."); Ex. 6, 82 ("Cast utilizes … ***receiver-implemented queueing***.").

As noted in the above admissions, the Receiver's local queue is kept in sync with a "remote queue" stored in the cloud (or "cloud queue") by virtue of the Receiver retrieving or loading segments of the remote queue into its local queue. *See also, e.g.*, Ex. 7, 82 (showing "Receiver" will "[r]etrieve remote queue" at start of a Cast session); G.Ex. 8, 67 ("Tells the receiver to load the cloud queue."); Ex. 5, 66 ("The ***receiver maintains a session for queue items*** until the last item completes playback … or until a sender ***loads a new queue on the receiver***."); Ex. 8, 74 (explaining that Cast's "[q]ueuing" functionality provides "[s]upport of Google's … cloud queue implementation so externally stored and created ***queue*** can be ***directly loaded into Cast devices***.").

Similarly, the Sender has a local queue that syncs with the remote queue when in a Cast session with a Receiver. *See, e.g.*, Ex. 9, 43 ("[YouTube Music Sender's] MDx integration includes a ***local queue*** that is kept in sync with the remote queue."). In this way, the Receiver and Sender "share a queue state" when in a Cast session. Ex. 10, 90 ("While Casting, YTM [Senders] and Cast receivers share a queue state via MDx's 'remote queue'.").

Although the Receiver's local queue is implemented differently when Casting YouTube versus GPM, both implementations infringe limitation 13.5(a).

a. Casting YouTube Involves a "Local Playback Queue"

To initiate a Cast session with a Receiver, the Sender transmits a "setPlaylist" message to an "MDx session server" that in turn sends a setPlaylist message to the Receiver. Ex. 1, ¶¶59-61. The setPlaylist message often includes a "videoId" corresponding to the multimedia item (e.g., song or video) that the Receiver is to begin playing. *Id.* In response to receiving a setPlaylist message with a videoId, the Receiver loads the videoId in its queue, thereby updating its internal logic such that the videoId corresponds to the item the Receiver is set to currently playback. *Id.*, ¶62.

By virtue of the Sender transmitting the setPlaylist message to the MDx server, the Sender also causes a "WatchNext" server to transmit a "WatchNextResponse" (WNR) to the Receiver that gets stored by the Receiver. *Id.*, ¶63. The contents of the WNR may vary depending on the Sender's playback state when the user initiated the Cast session, but the WNR typically will include at least a videoId for the item that the Receiver is set to currently playback and a videoId for the next item that the Receiver is set to playback (the WNR often includes additional videoIds, such as one for the item that comes before the current item). *Id.*, ¶64; G.Ex. 6, 91 ("When [Receiver] plays a video in the queue, the Watch Next service will send the video id … for the next video in the queue so that [Receiver] knows what to play when the current video finishes.").

Each of the videoIds contained in a WNR gets loaded into the Receiver's queue and corresponds to a "playback scenario" that the Receiver might perform. Ex. 1, ¶65. In other words, the videoIds from the WNR enable the Receiver to playback a given media item when the Receiver is to perform a given scenario. *Id.* For example, in a case where a Sender was playing the third video of a five-video playlist when a Cast session was initiated, the WNR can include a videoId for the current video (video #3), a videoId for the video before the current video (video #2), and a videoId for the video after the current video (video #4). *Id.*, ¶66. If the user provided skip-backward input at the Sender, the Receiver would retrieve from its queue the videoId for the video before the current video (video #2) to use to retrieve that video for playback. *Id.*, ¶67. Likewise, if the user provided skip-forward input at the Sender, or allowed the Receiver to complete playback of the current video, the Receiver would retrieve from its queue the videoId for the video after the current video (video #4) to use to retrieve that video for playback. *Id.*

Accordingly, when a YouTube Sender Casts to a Receiver, the Sender functions to cause the MDx and WatchNext servers to add to the Receiver's memory videoIds that at least dictate what the Receiver is set to playback currently and playback next. Ex. 11, 136:6-21, 151:10-17, 175:1-5. These stored videoIds form a "local playback queue" on the Receiver. Ex. 1, ¶¶69-71. More specifically, according to its plain and ordinary meaning, a "local playback queue" is a data construct on the playback device that can contain one or more resource locators (e.g., videoIds), each corresponding to multimedia content (e.g., a particular song or video) that the playback device

is to playback. *Id.* With respect to Google's Receiver, the data construct storing the current videoId alone and/or in combination with the data construct storing the next videoId satisfies the "local playback queue" of limitation 13.5(a). *Id.*, ¶¶70-71. As such, Google itself recognizing that its Receiver has a local queue, as discussed above, is not surprising. *Supra* §III.A.1.

To avoid this conclusion, Google argues that a "playback queue" is limited to an "ordered list of multimedia items selected by the user for playback."[3] Yet, even under this erroneous construction, the "playback queue" of limitation 13.5(a) is still satisfied.

To start, the Receiver's storage of a WNR literally satisfies the "playback queue" element under Google's construction and under the plain and ordinary meaning. Ex. 1, ¶¶71, 75, 78-81. Indeed, despite Google downplaying the significance of a WNR to a Receiver's playback (G.Br., 9-10), a WNR often contains multiple videoIds that, together, dictate the Receiver's playback of multiple items. As explained, a WNR can specify, for example, the item that the Receiver is currently set to play, the item that the Receiver is to play if the user decides to skip backwards, and the item that the Receiver is to play if the user decides to skip forwards or after the Receiver finishes playing the current item. *Supra* §III.A.1.a. Each of these items can be (i) from a collection of items directly selected by the user for playback or (ii) a service-recommended item that was indirectly selected by the user by virtue of the user's initial selection of an item or collection of items seeding the recommendation. Ex. 1, ¶¶59, 78-80. In this way, the WNR is a data structure that can provide multiple videoIds and specify the order in which the Receiver is to playback the corresponding multimedia content that was selected by the user. *Id.*, ¶¶75. Thus, Google's argument that the WNR does not amount to a "playback queue" apparently because it is not "a data structure [that] link[s] together different multimedia items" (G.Br., 9-10) is simply not true. Ex. 1, ¶¶101-102.

Moreover, while Google argues that the Receiver's data variables containing the videoIds of the current and next items are not an "ordered list" under its construction of "playback queue," these variables still infringe at least under DoE or at least present a genuine dispute of material fact precluding summary judgment. As an initial matter, Google incorrectly contends that Sonos surrendered any right to equivalents of the "playback queue" merely by virtue of adding the term

[3] The evidence demonstrates that Google's proposed construction for "playback queue" is contrary to how Google itself refers to the concept of a playback queue. Ex. 1, ¶¶73-74.

during prosecution. G.Br., 10-11. ***First***, the amendment had nothing to do with the particular ***form*** of the "local playback queue," which is the equivalent in question here. At most, the surrendered subject matter is any equivalent ***broader*** than the amendment – namely, "causing one or more cloud servers to add multimedia content to a local playback queue on the particular playback device." *See Insituform Techs., Inc. v. CAT Contracting, Inc.*, 385 F.3d 1360, 1368 (Fed. Cir. 2004) (holding that "plaintiffs have rebutted the Festo presumption that a narrowing amendment made for a reason of patentability surrenders the entire territory between the original claim limitation and the amended claim limitation."). ***Second***, the rationale underlying the amendment bears no more than tangential relation to the equivalent in question here, which relates only to the particular ***form*** of the "playback queue." *See Bio-Rad Lab'ys, Inc. v. 10X Genomics Inc.*, 967 F.3d 1353, 1365-66 (Fed. Cir. 2020) (affirming finding of no prosecution history estoppel because the claim amendment bears no more than tangential relation to the alleged equivalent). In fact, in making the amendment, Sonos made no statement or prior-art distinction regarding the "playback queue."

Turning to the substance of Sonos's DoE theory, a POSITA would readily appreciate that there is merely an insubstantial difference between a Receiver having (i) a singular data structure that provides an ordered list of the current and next videoIds for playback and (ii) a combination of data variables that collectively provides current and next videoIds that the Receiver knows to use for playback in a specific order. Ex. 1, ¶76. Indeed, "[t]o allow [Google] to escape infringement simply because it used separate [videoId data variables in a specific order], as opposed to a single [ordered list of videoIds], is the exact type of injustice the doctrine of equivalents prevents." *See Miles Labs., Inc. v. Shandon Inc.*, 997 F.2d 870, 877 (Fed. Cir. 1993); *see also Intel Corp. v. U.S. Int'l Trade Comm'n*, 946 F.2d 821, 832 (Fed. Cir. 1991) ("[W]e held that infringement under [DoE] may be established even though the accused device requires a number of components to perform functions which the patented invention achieves by use of one component.").

In this respect, regardless of whether a Receiver has a singular data structure or a set of data variables that provides videoIds for playback in a particular order, the Receiver performs substantially the same function (e.g., adding multimedia content to the "local playback queue"), in substantially the same way (e.g., by adding a current and next videoId to respective locations in the

Receiver's memory), to achieve substantially the same result (e.g., maintaining a "local playback queue" that includes videoIds that the Receiver knows to use for playback in a specific order) as a Receiver having a singular data structure that provides an ordered list of videoIds of the current and next media items for playback. Ex. 1, ¶77. Moreover, regardless of whether a Receiver has a singular data structure or a set of data variables that provides videoIds for playback in a particular order, the Sender performs the same function (e.g., causing playback of the corresponding media items to be transferred), in the same way (e.g., by transmitting a setPlaylist message to the MDx session server), to achieve the same result (e.g., causing the MDx and WatchNext servers to add the current and next videoIds to the Receiver's memory such that Receiver knows the order in which it should use the videoIds for playback). *Id.*

b. Casting GPM Involved a "Local Playback Queue"

Because the details of how a Sender Casts YouTube to a Receiver differ from how a Sender Casts GPM to a Receiver, Google makes a similar, but not identical, argument for non-infringement with respect to GPM. Google's GPM argument fares no better than its YouTube argument.

To initiate a Cast session with a Receiver, a GPM Sender transmitted a "Load" message causing the Receiver to load a current media item and receive from a "CQ server" a window of media items in an "ItemWindowResponse" (IWR) from a cloud queue at the CQ server. Ex. 1, ¶¶82-84. An IWR included a URL for (i) the current item, (ii) the item before the current item, and (iii) the item after the current item, which were all loaded into the Receiver's queue and dictated what item the Receiver would playback and when. *Id.*, ¶84. For instance, if the user provided input at the GPM Sender to play the song before the currently playing song, the Receiver would retrieve from its queue the URL corresponding to the "previous" song and use it to retrieve the corresponding song from a Bandaid cloud server. *Id.*, ¶85.

Thus, under the plain and ordinary meaning of "playback queue," as well as under Google's construction of that term, limitation 13.5(a) is satisfied by an IWR that was added to the Receiver's memory, which contained URLs for three multimedia items (i.e., songs) that were selected by the user for playback. *Id.*, ¶86. Like the WNR in the YouTube context, an IWR in GPM specified the song that the Receiver was currently set to play, the song before the current song, and the song after

the current song. *Id.* In this way, an IWR was a data construct that contained multiple resource locators (e.g., URLs) and specified the order in which the Receiver was to playback the corresponding multimedia content, thereby amounting to a "playback queue." *Id.*

c. Google's "Queue" Arguments Fail

Despite the clear recognition by Google itself, Google's attorneys throw out a variety of arguments as to why Casting does not involve a queue on the Receiver, all of which fail or at least demonstrate genuine issues of material facts, thereby precluding summary judgment.

First, at the core of Google's argument is the false premise that a "local playback queue" and a "cloud queue" are mutually exclusive. G.Br., 1, 3-7, 13-15. In other words, Google incorrectly contends that a "playback queue" can only exist at one device in a multi-device system. The flaw in this premise is clear from the '615 Patent and Google's own words. Ex. 1, ¶88.

There is nothing in claim 13 that precludes the possibility that some other queue might exist in the system beyond the claimed "local playback queue." *Id.*, ¶89. In fact, the '615 Patent makes it clear that multiple devices in a system can share a queue such that each of multiple devices can maintain a respective copy of the queue (or some portion thereof). *Id.*, ¶¶90-92. For instance, the '615 Patent describes embodiments where (i) a playback device's "local playback queue" is kept "synchronized" with an "application-specific queue" maintained at a control device ('615 Pat., 16:20-31), (ii) a playback device "periodically fetches a short list of tracks to play next" from an "application-specific queue" that are then loaded into the playback device's "local playback queue" (*id.*, 16:63-17:1), and (iii) a playback device's "local playback queue" and a control device's "application-specific queue" are synchronized to a "shared queue [ ] provided between" the two (e.g., a queue in the cloud). *Id.*, 16:63-17:4.

As discussed before, despite Google's YouTube and GPM implementations utilizing a "cloud queue," Google itself describes multiple devices within its system as sharing a queue much like the '615 Patent's teachings. *Supra* §III.A.1. Moreover, the Receiver's receipt of a WNR in the YouTube context and an IWR in the GPM context are very similar to the '615 Patent's teachings regarding a playback device periodically fetching a set of tracks from a queue maintained at another device that are then loaded into the playback device's "local playback queue." Ex. 1, ¶¶93, 96.

Second, for YouTube, Google asserts that a Receiver only "request[s] cloud queue items one-by-one," and thus, does not have a local queue. G.Br., 5; *id.*, 1, 7. This assertion is contradicted by the evidence. As noted above, a WNR includes multiple videoIds corresponding to multiple items from the cloud queue, such as the respective videoIds of the item before and the item after the current item. *Supra* §III.A.1.a.; Ex. 1, ¶¶95-96. Google's 30(b)(6) witness also testified that there are circumstances where the first WNR provides the Receiver with videoIds of the first two items it is to playback. *See* Ex. 11, 161:5-162:1, 163:4-164:1.

Third, for both YouTube and GPM, Google argues that, by virtue of there being a cloud queue, the Receiver does not locally store the "***complete***" set of the media items that the Receiver is scheduled to playback and thus, cannot have a "local playback queue." *See* G.Br., 8-9, 15. But this implied requirement is contrary to the '615 Patent that, for example, explains that a playback device "periodically fetches a short list of tracks" from a queue maintained at another device that are then loaded into the playback device's "local playback queue." Ex. 1, ¶¶97-98. Google's "complete" requirement is also contrary to how Google itself describes the different queues that exist alongside the cloud queue in its own system. *Id.*, ¶¶54-57, 99.

Fourth, Google argues that the claimed "playback queue" must be able to be "managed and edited by the user" and "[t]he local variables that Sonos accuses cannot" (G.Br., 8-9). This is another limitation Google is reading into "playback queue" (no such requirement is found in the claims or patents), and also a red herring. While Casting, a user can indisputably "manage" the queue via the Sender in various ways, including modifying what media item the Receiver is to play next. Ex. 1, ¶¶103-107. In doing so, the Receiver's data construct amounting to the "local playback queue" is updated to reflect the change in the media item that is to play next. *Id.* At least in this way, the Receiver's "local playback queue" can be managed by the user.

Fifth, for YouTube, Google argues that the Receiver's next videoId data variable is limited to a "recommended 'autoplay' video" or song (*see* G.Br., 7-8) and this data variable "exists for only a few milliseconds …." G.Br., 9. Neither point is correct. Indeed, Google's 30(b)(6) witness explained that the Receiver will store in a data variable the videoId of an item that the Receiver is to play next, whether or not that item is a service-recommended item (e.g., an "Autoplay" video)

or from a collection of items directly selected by the user.[4] Ex. 1, ¶109; Ex. 11, 174:10-175:5 ("[T]he next one doesn't have to be associated with the autoplay."). Moreover, far from existing for only a few milliseconds, Google's 30(b)(6) witness explained that the Receiver stores the "next" data variable until it finishes playback of the current item, after which "it will look up that data model and then it will extract the appropriate endpoint for the next video to play." *Id.*, 175:6-23.

Sixth, for YouTube, Google incorrectly suggests that the Receiver only retains a videoId for a service-recommended media item "well after playback has been transferred." G.Br., 9. In truth, there are numerous cases in which the second media item that the Receiver is to play will be a service-recommended media item such that the Receiver's first WNR will contain a videoId for a service-recommended media item as the next videoId. Ex. 1, ¶110. Even outside of those cases, the first WNR stored at the Receiver often contains a videoId for a service-recommended media item that will be used "after the device has exhausted the playlist" (G.Br., 9). Ex. 1, ¶110.

2. "Multimedia Files" Need Not Be Added to a "Playback Queue"

Google uses another false premise to support its non-infringement position by contending that "add[ing] multimedia content to a local playback queue" must involve adding a "multimedia file." G.Br., 4, 11-12, 15. To make this argument, Google distorts the plain language of limitation 13.5(a) – "causing one or more first cloud servers to add multimedia content to a local playback queue … , **wherein** adding the multimedia content to the local playback queue **comprises** the one or more first cloud servers adding, to the local playback queue, one or more resource locators" – in a manner contrary to how any POSITA would interpret it. Ex. 1, ¶¶111-17.

Limitation 13.5(a) follows a standard claim-drafting convention in which the "wherein" clause specifies ***how*** the "one or more first cloud servers" are to "add multimedia content to a local playback queue." *Id.*, ¶113. The "wherein" clause explains that "add[ing] multimedia content to a local playback queue" means, and is characterized by, "adding, to the local playback queue, one or more *resource locators*." *See, e.g.*, *Griffin v. Bertina*, 285 F.3d 1029, 1033 (Fed. Cir. 2002) ("Each 'wherein' clause … giv[es] meaning and purpose to the [claimed functions]."); MPEP § 2111.03

[4] Google's suggestion that a recommended media item is not "selected by the user" (G.Br., 9) is misplaced because such a media item is identified by Google's service based on the user's selection and thus, is indirectly "selected by the user." Ex. 1, ¶79.

("[T]ransitional term 'comprising' … is synonymous with … 'characterized by[.]'").

Moreover, when viewing claim 13 as a whole, no POSITA would interpret limitation 13.5(a) as Google does because it would result in non-sensical redundancy. Ex. 1, ¶114. In this regard, limitation 13.6 later recites "the particular playback device ***retrieving the multimedia content from one or more second cloud servers*** of a streaming content service and playing back the retrieved multimedia content." A POSITA would readily appreciate that, if "adding the multimedia content to the local playback queue" of limitation 13.5(a) was interpreted to require adding one or more multimedia files to the "local playback queue on the particular playback device" as Google claims, there would be no need for the "playback device" to "retriev[e] the multimedia content ***from one or more second cloud servers***," as required by limitation 13.6, because it would already have the multimedia content locally in its "playback queue." *Id.* Thus, Google's interpretation is contrary to the plain meaning that a POSITA would ascribe to limitation 13.5(a).[5]

The '615 specification and file history also demonstrate that Google's interpretation is contrary to the plain and ordinary meaning. *Id.*, ¶¶115-16. In this regard, the '615 Patent demonstrates to a POSITA that a "resource locator," such as a URL or other identification, is first added to the playback device's "playback queue" before it retrieves the music corresponding to the "resource locator," which is exactly what limitations 13.5(a) and 13.6 recite. *See, e.g.*, '615 Pat., 11:62-12:3, 12:53-63. In fact, in the '615 file history, Sonos explained to the USPTO that limitation 13.5(a) "recite[s] 'causing one or more first cloud servers to add the multimedia content to a local playback queue on the particular playback device' ***by 'adding***, to the local playback queue, one or more resource locators ….'" D.I. 185-8, App'x B, 4 (original emphasis omitted).

Thus, Google's non-infringement argument for both the YouTube apps and the GPM app premised on the assertion that limitation 13.5(a) requires adding a "multimedia file" to the "*local playback queue*" is contrary to the intrinsic evidence and thus, fails.

3. Casting YouTube Adds "Resource Locators"

Limitation 13.5(a) specifies that "adding the multimedia content to the local playback queue" is accomplished by adding "one or more resource locators ...." Google raises a specific

[5] Google's discussion of claim 20 is a red herring as that claim merely recites a specific type of "resource locator" that is added to the "local playback queue." D.I. 184, 18-19; D.I. 202, 13.

argument for YouTube: a videoId does not amount to a "resource locator." Google is wrong.

To facilitate Casting from a YouTube Sender to a Receiver, Google's MDx and WatchNext servers are caused to add one or more videoIds to the Receiver's "local playback queue." Ex. 1, ¶¶61-63. Each videoId enables the Receiver to access (e.g., locate and retrieve) a given media item from a Bandaid server. *Id.*, ¶¶68, 119. Indeed, much like certain URLs, the Receiver provides a videoId to an intermediate service (the Player Service) that translates the videoId into at least one URL for retrieving the given item from a Bandaid server. *Id.*, ¶119. Consequently, the "resource locator" element is satisfied by the videoId under the plain and ordinary meaning of that term in the context of the '615 Patent because it is information that enables the Receiver to access a resource (e.g., a song or video) in the cloud. *Id.*; *cf. ACTV, Inc. v. Walt Disney Co.*, 346 F.3d 1082, 1092-93 (Fed. Cir. 2003) (explaining the term "URL" is met "so long as it provides sufficient information for the system to identify a [resource]" and finding "file names that identify specific information stored on … servers" amounted to URLs). In fact, in its *Markman* brief, Google conceded that information that "point[s], ***indirectly***, to the location of a resource" amounts to a "resource locator" (*see* D.I. 200, 21), which is ***exactly*** what a videoId does. Thus, it is unsurprising that one of Google's engineers referred to the videoIds as "pointers" to content that the Receiver could play. Ex. 1, ¶119 citing Ex. 12, 108:5-10, 114:19-115:1.

Moreover, the "resource locator" element is satisfied literally, and under DoE, even under Google's overly-narrow construction: an address of a resource on the Internet. As to literal, after the Player Service translates a given videoId into at least one URL, that URL is added to the Receiver's memory in a "DashManifest" that is used to retrieve the corresponding media item from a Bandaid server for playback. *Id.*, ¶120. Google cannot dispute that a URL amounts to "an address of a resource on the Internet." Thus, the "resource locator" element is literally satisfied under this alternative infringement theory with the "DashManifest" storing the URL constituting the "local playback queue" since it is a data construct on the Receiver that contains a resource locator (e.g., URL) corresponding to multimedia content the Receiver is to playback. *Id.*, ¶121.

As to DoE, Dr. Schmidt explained that there is merely an insubstantial difference between a Sender (i) causing one or more cloud servers to add at least one "address of a resource on the

Internet" to a "local playback queue" and (ii) causing one cloud server to first add at least one videoId to a "local playback queue" and then another cloud server to subsequently translate the videoId into an "address of a resource on the Internet" that the Receiver uses for playback. *Id.*, ¶122. *See EMI Grp. N. Am., Inc. v. Intel Corp.*, 157 F.3d 887, 896 (Fed. Cir. 1998) ("Equivalency is not defeated by using an additional step to achieve what the patentee does in one step."). In this respect, whether a Sender directly or indirectly causes one or more cloud servers to add at least one "address of a resource on the Internet," the Sender is performing substantially the same function (e.g., causing one or more cloud servers to add multimedia content to the Receiver's "local playback queue"), in substantially the same way (e.g., by sending a message to the one or more cloud servers that causes this function to occur without any intervening user action), to achieve substantially the same result (e.g., one or more cloud servers adding at least one "address of a resource on the Internet" to the Receiver's memory to facilitate playback). Ex. 1, ¶122. In other words, there is merely an insubstantial difference between a cloud server adding an "address of a resource on the Internet" in one step and a cloud server adding information representing an "address of a resource on the Internet" in a first step that is resolved into that address in a subsequent step. *See EMC Corp. v. Pure Storage, Inc.*, 154 F. Supp. 3d 81, 96 (D. Del. 2016) (denying non-infringement MSJ because "jury could reasonably conclude that ***returning an index representing the identifier*** either literally satisfies the claim language '***returning the identifier***' or satisfies it under [DoE].").

B. Claim 13 of the '615 Patent Is Not Invalid Over the Alleged Prior Art

To continue the false narrative that it was Google (and not Sonos) who was the innovator, Google concocts an anticipation theory based on an alleged November 9, 2010 version of its own product—the YouTube Remote ("YTR") software app—by blatantly ignoring and/or misinterpreting limitations of claim 13 of the '615 Patent. When all the limitations are properly considered, it is clear that the YTR app is missing multiple features of the claim.

Recognizing the deficiencies of its anticipation theory, Google then asserts that claim 13 would have nevertheless been obvious over the YTR application in view of an allegedly related Google patent—U.S. Patent No. 9,490,998 (the "'998 Patent")—and various other secondary references that allegedly evidence the "general knowledge" of a POSITA. But Google's conclusory

and overly simplistic obviousness theories also fail because the secondary references do not disclose many of the features Google relies on them for, and Google's YTR app teaches away from making any such combinations or modifications in the manner Google proposes.

What's more, Google's own development story for the YTR app confirms that there is no anticipation and that claim 13 would not have been obvious at the time of the invention because Google did not allegedly ***add*** the missing features of claim 13 to a version of the YTR app until ***after*** the July 15, 2011 invention date.[6] Moreover, in order to address multiple complex and distinct issues in its motion, Google attempts to circumvent the Court's strict page limits and prove invalidity by focusing on only some, but not all, of the limitations of claim 13, and by not attaching evidence upon which it relies as exhibits. To prove invalidity, however, Google is required to prove by clear and convincing evidence that ***each and every*** limitation is disclosed by or rendered obvious over the prior art. Thus, the Court should deny Google's motion for this reason alone.

Finally, as explained below, there are, at the very least, genuine issues of material fact regarding the validity of '615 claim 13 that preclude the granting of Google's motion.

1. Overview of Google's YTR App

The first version of Google's YTR app was allegedly released on November 9, 2010. *See* G.Br., 16. This "beta" version of the YTR app could allegedly be installed on an Android mobile phone and "paired" in a "session" with a TV or computer so that the YTR app could be used to control the playback of videos on such a device. *See id.*, 18; G.Ex. 1, ¶132; Ex. 1, ¶125-27. As explained by Google, "in order for a YTR [app] to be paired to one or more televisions … over the Internet," the user had to navigate to the "Leanback" website and "log[] each television into the same YouTube account that the YTR [app] is logged into." G.Br., 18. In this way, the architecture that enabled such "pairing" (and subsequent communication between the paired devices) relied on an intermediary cloud server referred to by Google as a "Lounge Server" or "MDx server." *See id.*; G.Ex. 1, ¶¶128, 132. Herein, Sonos refers to a paired TV or computer as a "Leanback Screen." *Id.*

One of the touted benefits of this cloud-based architecture was that the mobile phone running the YTR app (herein "YTR device") did not have to be connected to the same local area

[6] For purposes of summary judgment, Google is not challenging the invention date. *See* G.Br., 19 n.7; G.Ex. 1, ¶124.

network (LAN) (e.g., a home Wi-Fi network) as the Leanback Screen. As such, the YTR app could be paired with and control a Leanback Screen when the YTR device was only connected to a wide area network (WAN) (e.g., a 3G cellular network). *See* Ex. 14, 94:2-21; GOOG-SONOSNDCA-00071320, 2:51-3:20 (YTR app can be paired with Leanback Screen "***over 3G and Wi-Fi***"); G.Ex. 1, ¶157 ("In order to pair … the YTR [app] and a Screen both had to be connected to the ***Internet***—which ***could be*** done by connecting to a user's home network through ***Wi-Fi***."); '998 Pat., 4:51-55 ("[B]y using the network service as an intermediary, the remote control and the controlled device … may ***not need to be connected to the same local area network*** …."); Ex. 1, ¶128.

While Google alleges that a YTR app could be simultaneously paired with multiple Leanback Screens in a session, Google acknowledges that in such a configuration the same media would be played on ***all*** the Leanback Screens and that the YTR app and/or cloud-based Lounge Server would be configured to control the playback on all paired Leanback Screens ***collectively*** (as opposed to controlling any one of the multiple Leanback Screens ***individually*** or controlling a subset of the multiple Leanback Screens). *See* G.Ex. 1, ¶¶140, 144, 158, 160-162; Ex. 1, ¶127.

2. The YTR App Does Not Anticipate Claim 13

a. The YTR App Does Not Disclose Limitation 13.2

Limitation 13.2 requires "after connecting to a [LAN] via a network interface, identifying playback devices connected to the [LAN]." The plain language requires the "control device" to not only ***identify*** playback devices, but also to identify playback devices that are connected to the ***same LAN*** as the control device. Ex. 1, ¶146. Moreover, when read in light of limitation 13.4, limitation 13.2's "identifying" must allow a particular playback device (from the identified playback devices) to be selected via the control device to transfer playback of multimedia from the control device to the particular playback device. *See id*. The YTR app does not teach this limitation.

According to Google, the YTR app performs the "identifying" of limitation 13.2 because "[e]ach time a television was paired [with a YTR app], it registered with the MDx server and the server sent a 'loungeScreenConnected' message to the YTR [app] indicating that the television was connected and available to transfer playback." G.Br., 18. Google is wrong. Conspicuously omitted from Google's brief is the actual description of the "loungeScreenConnected" message from the

document Google cites, which explains that this message merely "informs the [YTR app] that there is at least one screen connected in the session."[7] Ex. 13, 37. Notifying the YTR app that there is "at least one screen … in the session," and doing so via a message sent from the ***cloud-*** or ***WAN-***based Lounge Server/MDx server, does not enable the YTR app to ***identify*** the Leanback Screen in the session, let alone identify the Leanback Screen as being connected to the ***same LAN*** that the YTR device is connected to, as required by limitation 13.2. *See id.*; Ex. 1, ¶147.

Further, even if the loungeScreenConnected message included information from which the YTR app could identify the specific Leanback Screen that was registered with the ***WAN***-based Lounge Server (there is no such evidence), this would still not satisfy limitation 13.2 because the YTR app could not identify the Leanback Screen as being connected to the ***same LAN*** that the YTR device is connected to. Ex. 1, ¶148. This makes sense because the YTR app receives the loungeScreenConnected message from the ***WAN***-based Lounge Server regardless of whether the YTR app and Leanback Screen are both operating on the same LAN or different LANs, or whether the YTR app is only operating on a WAN (e.g., a 3G cellular network), and the loungeScreenConnected message provides no indication of the network that the Leanback Screen is connected to. *See supra* III.B.1.

b. The YTR App Does Not Disclose Limitation 13.4

The plain language of limitation 13.4 requires at least ***two separate and distinct inputs*** to transfer playback from the control device to a particular playback device, where each input must meet the specific requirements set forth in limitation 13.4's subparts (i) and (ii). Ex. 1, ¶150. The YTR app does not teach the two required inputs of this limitation.

According to Google, the YTR app detects the inputs of limitation 13.4 via selection of the "menu" and/or "Connect" buttons shown in the image here. *See* G.Br., 19. Again, Google is wrong.

[7] The term "Connected" in the message name refers to the Leanback Screen being registered with and connected to the cloud-based Lounge Server and thus, in a "session" with a YTR app – it does ***not*** refer to a connection between the Leanback Screen and whatever network the Leanback Screen and/or YTR device is operating on. *See* Ex. 13, 37; Ex. 1, ¶147 n.27.

Selection of the "menu" button does not meet either of the required inputs at least because it is not a button for transferring playback from the YTR app or a button for selecting a particular playback device to transfer playback to. Ex. 1, ¶152. Instead, as shown in the image above, the "menu" button does exactly what its name describes, it simply activates a "menu" on the YTR app, where the "menu" includes a "Refresh," "Switch User," "Search," and "Connect" button. *Id.*

The only other input identified by Google is the input received via selection of the "Connect" button. *See* G.Br., 19. However, a ***single*** input via the "Connect" button cannot satisfy the ***two*** separate and distinct inputs of limitation 13.4. *See In re Robertson*, 169 F.3d 743, 745 (Fed. Cir. 1999) (reversing finding of anticipation because the cited reference only disclosed two of the claimed fastening elements and did not disclose the third claimed fastening element, which "is separate from and in addition to the other mechanical fastening means and performs a different function than they do."); Ex. 1, ¶153. Thus, the YTR app fails to teach limitation 13.4.

Even if a single input could satisfy the two separate and distinct inputs of limitation 13.4 (it cannot), a selection of "Connect" is not "a selection of ***the particular playback device*** from the identified playback devices connected to the [LAN]." Ex. 1, ¶154. For instance, as shown in the image above, the "Connect" button does ***not*** provide any indication of ***the particular Leanback Screen*** to transfer multimedia to. This makes sense because, as discussed for limitation 13.2, the YTR app did not even identify the specific Leanback Screens paired in a session with the YTR app. Without such an identification, the YTR app could not present a particular Leanback Screen for selection or detect its selection.

Moreover, Google's expert confirms that a selection of the "Connect" button is ***not*** "a selection of ***the particular playback device*** from the identified playback devices connected to the [LAN]." According to Dr. Bhattacharjee, when multiple Leanback Screens are paired in a session with a YTR app, "[p]ressing the Connect button transfers playback to ***all the Screens*** that have been paired with the YTR [app] in a session …." G.Ex.1, ¶162. In other words, from the perspective of the YTR app, a selection of the "Connect" button merely indicates a selection of ***all*** Leanback Screens in a session. Ex. 1, ¶155. A selection of a button divorced from any particular Leanback Screen is ***not*** a selection of "***the particular***" Leanback Screen to transfer playback to.

Further, Google's own product development story confirms that the alleged November 9, 2010 prior art version of the YTR app is missing features of limitation 13.4. Specifically, Google acknowledges that it was not until a later ***non-prior art foreign*** version of the YTR app that Google first "released" the ability to "select and control" "individual" Leanback Screens. *See* G.Br., 20.[8]

c. Google Failed to Prove YTR Discloses Limitations 13.5-13.6

Google fails to meet its burden to prove by clear and convincing evidence that the YTR app discloses limitations 13.5-13.6. Google's arguments for at least certain aspects of these limitations rely on an "API" document (G.Ex. 10) and/or source code that allegedly describe the November 9, 2010 version of the YTR app. *See* G.Br., 17-18. However, the "API" document is dated July 12, 2010 and the "evidence" cited by Google tying it to the November 9, 2010 YTR app is an uncorroborated 2022 declaration of an interested witness, Janos Levai, who did not even work at Google until July 2011 (as an intern), which is insufficient corroboration. G.Ex. 11, ¶¶2, 9; *Union Carbide Chems. & Plastics Tech. Corp. v. Shell Oil Co.*, 308 F.3d 1167, 1189 (Fed. Cir. 2002) ("Uncorroborated oral testimony by interested parties is insufficient as a matter of law to establish invalidity of [a] patent.") (citation omitted); *Juicy Whip, Inc., v. Orange Bang, Inc.*, 292 F.3d 728, 743 (Fed. Cir. 2002) (evidence presented by accused infringer was "insufficient as a matter of law to surmount the clear and convincing evidence hurdle" because "[t]he testimony about the [alleged prior art] came more than eight and twelve years, respectively, after the witnesses saw the [alleged prior art]."); Ex. 1, ¶134. Likewise, Google's "evidence" tying the source code to the November 9, 2010 YTR app is the same uncorroborated declaration of Mr. Levai. G.Ex. 11, ¶6.

Moreover, limitation 13.5 requires the "transferring playback" to include "causing the playback at the control device to be ***stopped***." However, the November 14, 2010 video relied on by Google does not establish that the media on the YTR device was "stopped." *See* G.Ex. 1, ¶182. To the contrary, the media on the phone's screen appears to still be in a playback state (albeit paused) after the alleged "transfer" of playback.

3. The YTR App Does Not Render Obvious Claim 13

Google's conclusory and overly simplistic obviousness theories fail for numerous reasons.

[8] Google asserts that this added functionality was included in a 12/1/2011 capture of source code, but that is irrelevant as it is also after Sonos's uncontested 7/15/2011 invention date. G.Ex. 1, ¶170.

For instance, contrary to Google's assertions (*see* G.Br., 19-21), the limited evidence Google cites for the '998 Patent, the Tungsten system, the Al-Shaykh Publication, and Sonos's alleged prior art products does not disclose any "inputs to ***transfer playback*** from the control device to a particular playback device," let alone an input in the form of "a selection of the particular playback device from the identified playback devices connected to the [LAN]," as required by limitation 13.4. *See* Ex. 1, ¶160. Instead, Google's evidence for these references merely shows that a control device could be used to ***control playback*** on one or more playback devices (e.g., starting or stopping playback) or for ***streaming media content*** from the control device to a playback device. *Id.*[9] Thus, a POSITA would not have been motivated to consider the teachings of these alleged prior art references in order to modify the YTR app to include the "inputs to ***transfer playback***" required by limitation 13.4. *Id.*. These references do not include such teachings.

Take for example Google's lead obviousness argument based upon the '998 Patent's disclosure at 10:62-11:6. G.Br., 20. That disclosure does not mention or suggest "***transferring*** playback" from a "remote control" to a "controlled device." Ex. 1, ¶171-72. Consequently, this disclosure does not teach the selection of a particular "controlled device" to ***transfer*** playback to.

Moreover, it is not clear that this disclosure even teaches the ability of a "remote control" to present "controlled devices" in a manner that would allow a user to select a particular "controlled device" for individual control, as Google asserts. Ex. 1, ¶173; Ex. 15, 104:21-105:3 (after inventor testified about what she allegedly was "thinking" about at the time, when pressed to explain what the "plain language" of the patent actually disclosed, she could not do so). Instead, consistent with how the alleged prior art YTR app operated, it appears that this passage is referring to the ability to control any and all "controlled devices" that have been "previously paired" with a "remote control" in a session, with no ability to choose from amongst those "controlled devices." Ex. 1, ¶173.

As another exemplary failure of Google's lead obviousness argument, the '998 Patent does

[9] "Transferring playback" from a "control device" to a "particular playback device" as recited in claim 13 is not met by merely streaming media content from a control device to a playback device and/or initiating playback on one or both devices. *See* limitations 13.4-13.6; Ex. 1, ¶160. The control device has to be capable of being in a playback state when it "detect[s] a set of inputs to transfer playback" and thereafter "caus[e] playback to be transferred from the control device to the particular playback device," which includes "causing playback at the control device to be stopped" and "causing the particular playback device to play back the multimedia content." *Id.*

not disclose a "control device" "identifying playback devices connected to the [LAN]," as required by limitation 13.2, and thus, cannot make up for the lack of this functionality in the YTR app. *Id.*, ¶175. While the '998 Patent discloses a ***cloud server*** maintaining and using "unique identifiers" for the "remote control" and "controlled devices" that are paired in a session ('998 Pat., 4:4-58, 8:1-10, 17:44-57), the '998 Patent does not disclose anything about the "***remote control***" receiving the "unique identifiers" of the "controlled devices" or otherwise identifying the "controlled devices" based on the "unique identifiers," let alone identifying the "controlled devices" as being connected to the same LAN as the "remote control." Ex. 1, ¶175. In fact, the '998 Patent shuns such a disclosure and instead touts a reliance on an intermediary cloud/WAN server—including its maintenance and use of the "unique identifiers"—as enabling the "remote control" and "controlled device[s]" to be paired in a session while "***not need[ing] to be connected to the same [LAN]***, nor in physical proximity to each other." '998 Pat., 4:51-55; Ex. 1, ¶176.

As part of Google's backup obviousness theory, Google points to an alleged 2010 version of Apple AirPlay. G.Br., 21. Google's reliance on Apple AirPlay is also misplaced. Unlike the YTR app, which was specifically designed with only a ***WAN***-based communication channel to allow the YTR app to be paired with and control a Leanback Screen that was ***not on the same LAN***, Apple AirPlay utilized a fundamentally different system architecture that required an AirPlay control device to be on the ***same LAN*** (Wi-Fi) as the AirPlay playback device. Ex. 1, ¶161. Thus, a POSITA would not be motivated to modify the YTR app, which allowed the YTR app to be paired with and control Leanback Screens on different networks, with a "***fundamentally different***" system like AirPlay, which requires the control device and players to be on the same network. Ex. 1, ¶161-63; Ex. 16, 96 (Google characterizing YTR as "fundamentally different" than Apple AirPlay); *see also* '998 Pat., 1:14-50 (distinguishing WAN-based system architecture of the YTR app from system architectures of other prior art systems like Apple AirPlay that enabled remote controls to communicate directly with playback devices over a LAN).

In addition to the above-described deficiencies based on the teachings (or lack thereof) in Google's secondary references, Google's obviousness theories also fail for many other reasons. For example, the system architecture for the alleged prior art version of the YTR app teaches away

from modifying the YTR app in the manner proposed by Google. Ex. 1, ¶164. Moreover, at the very least, the architecture makes such modification more difficult than Google asserts and would require changes that Google and Dr. Bhattacharjee fail to address in their conclusory analysis. *Id.* As explained by Google, "in order for a YTR [app] to be paired with one or more televisions … over the Internet," the user had to "log[] each television into the same YouTube account that the YTR [app] is logged into." G.Br. 18. In other words, when a user wished to control media playback on a particular Leanback Screen using the YTR app, the user identified the particular Leanback Screen (without caring about whether the Leanback Screen and YTR app were connected to the same LAN) and used the ***user interface of that particular Leanback Screen*** to log the Leanback Screen into the same YouTube account that the YTR app was logged into, which caused the WAN-based Lounge Server to pair the devices in a session. Ex. 1, ¶161. Given this architecture, there was no need for the YTR app to identify the Leanback Screen (let alone identify it as being on the same LAN as the YTR app), and then present the Leanback Screen for selection via ***the user interface of the YTR app*** because the user had already identified and selected the particular Leanback Screen they wished to control. *Id.* In this regard, the YTR app taught against detecting the claimed "selection" required by subpart (ii) of limitation 13.4. *Id.*

Moreover, in a scenario where a YTR app and multiple Leanback Screens were paired together in a session (the scenario Google relies on), Google acknowledges that the YTR app and/or Lounge Server were configured to control the playback of the same media on ***all*** paired Leanback Screens ***collectively***. G.Ex. 1, ¶¶140, 144, 158, 160-162. In other words, the YTR app and/or Lounge Server did not have the capability to control ***a particular*** paired Leanback Screen. Ex. 1, ¶165. Nevertheless, Google asserts that limitation 13.4 was obvious because a POSITA would have modified the YTR app to allow a user to select and control a particular Leanback Screen in a session (G.Br., 19-21), but Google has not adequately explained how a POSITA would have modified the system architecture such that, after multiple Leanback Screens have been paired together in a session by the WAN-based Lounge Server, the YTR application and/or Lounge Server would be configured to only control a particular Leanback Screen. *Id.* Further, a POSITA would not have been motivated to modify the YTR app to allow for a selection of a particular Leanback

Screen because using multiple Leanback Screens was not even a prominent feature for the YTR app. *Id.*, ¶166. Indeed, Google's own declarant testified that: (i) using YTR with multiple screens "sounds like an [edge] case," (ii) "I don't remember using multiple screens at the same time," and (iii) "why would I want to see a video on multiple screens?" Ex. 14, 110:23-111:16.

As yet another exemplary deficiency in Google's obviousness theories, despite the YTR app not disclosing limitation 13.2's "identifying playback devices connected to the local area network" (*supra* III.B.2.a.), Google fails to adequately explain how a POSITA would modify the YTR app to include the missing features of this claim limitation. Ex. 1, ¶168. Instead, Google declares that it would have been obvious without any analysis or support. G.Br., 18-21 (concluding that "[t]he YTR prior art satisfies these limitations [13.2 and 13.4], or they are obvious" but only analyzing obviousness for limitation 13.4); *see also* G.Ex. 1, ¶¶165-166, 173 (concluding it would have been obvious to modify the YTR application to "identify playback devices … on the [LAN]" because this was well known). The mere conclusion that it would have been obvious to modify the YTR app to include the features of limitation 13.2 is insufficient to prove invalidity. *See ActiveVideo Networks, Inc. v. Verizon Communications, Inc.*, 694 F.3d 1312, 1327 (Fed. Cir. 2012) (affirming JMOL overturning obviousness verdict because "the expert's testimony on obviousness was essentially a conclusory statement that a [POSITA] would have known, based on the 'modular' nature of the claimed components, how to combine any of a number of references to achieve the claimed inventions," which "is not sufficient and is fraught with hindsight bias.").

Further, Google's conclusory assertion that modifying the November 9, 2010 version of the YTR app to include the features of limitations 13.2 and 13.4 would have provided an obvious "improve[ment]" that was "straightforward" to implement in the YTR app (G.Br., 21) ignores the YTR system architecture, which, as explained above, both teaches away from Google's proposed modifications and also renders such modifications more complicated than Google posits. Ex. 1, ¶167. This assertion is also contradicted by Google's own development story for the YTR app. *Id.*. Indeed, it took Google ***more than a year*** – and ***after*** the '615 Patent's invention date – to modify the YTR app to allegedly allow a user to select a particular Leanback Screen for playback/control. G.Br., 20 (asserting that version of YTR with feature was released by 1/25/2012).

Clearly this was not a straightforward improvement. Ex. 16, 94-95 (Google acknowledging that modifying the YTR pairing architecture "requires a complete redesign" and "lots of changes").

Lastly, secondary considerations of non-obviousness for the claimed invention exist (or at a minimum, there is a genuine issue of fact regarding their existence) that undercuts Google's conclusory obviousness argument. Ex. 1, ¶177-79. Indeed, because Google infringes '615 claim 13, evidence such as praise of a Sender that is provisioned with the accused Cast technology has a sufficient nexus to the invention. *See, e.g., Callaway Golf Co. v. Acushnet Co.*, Case No. 06-CV-091, 2007 WL 4326776, at *1 (D. Del. Dec. 4, 2007) (holding evidence of praise of accused products is admissible where accused products infringed asserted claims).

C. Google Infringes Claim 1 of the '885 Patent

Google disputes only whether its speaker groups are "zone scenes" under Google's construction of the term. G.Br., 21-25. Google raises no noninfringement argument under the proper construction of "zone scene," which tracks the plain claim language: "a previously-saved grouping of zone players that are to be configured for synchronous playback of media when the zone scene is invoked." *See* D.I. 209.02, 8; *see also* D.I. 126, App. A, 27.

As an initial matter, Google's proposed construction, which adopts Judge Albright's prior preliminary oral construction, is not law of the case. Dkt. 184, 1-3. It is also incorrect. *See, e.g.*, *Sonos, Inc. v. Google LLC*, No. 3:21-cv-07559-WHA, D.I. 60, 11-16; *id.*, D.I. 66, 7-8.

Regardless, under Google's proposed construction, the Accused Google Players infringe because they are capable of being included in "a previously saved grouping of zone players according to a common theme." Each Accused Google Player contains program instructions for receiving a message indicating that it has been added to a speaker group. D.I. 209.02, 9 (describing "join_group" message with a "group uuid" and a "group name"). A user can name the speaker group according to anything the user desires, including, for example, names that specify a theme like a time of day ("Morning") or an area of the user's home ("Garden"). *See* G.Br. 24; D.I. 209.02, 10-11. These are the ***exact*** types of "themes" contemplated by the '885 Patent. *Id.*; '885 Pat., 8:47-61, 10:36-41, Fig. 5A. The Accused Google Players' undisputed capability to be included in previously-saved groups "according to a common theme" establishes infringement. *See Finjan,*

Inc. v. Secure Computing Corp., 626 F.3d 1197, 1204 (Fed. Cir. 2010).

Because Google cannot prevail under the plain language of the construction it has adopted, Google tries to further limit "according to a common theme" to require an "option to add or change 'scene' or 'theme' information" other than a group name, such as "a particular playlist, volume, equalization, or other 'attributes' described in the specification that are consistent with the common theme." G.Br. 22-23; *see also id.* at 24 (referring to zone scene "settings"). However, such "attributes" (or "'theme' information") – let alone the capability to "add or change" them – are not required by claim 1, even under Google's construction, and cannot be used to avoid infringement. Google's primary support for its new limitation is a passage in the specification explaining that devices "***can*** be configured in a particular scene … where … attributes for the grouping are automatically effectuated." *Id.*, 24 (emphasis changed); *see also* '885 Pat., 9:20-30 ("[A] zone scene command ***could*** apply [] attributes"). But this is plainly just one example embodiment of a "zone scene," and when describing other "zone scene" embodiments, the specification makes no mention of these other "attributes." *See e.g.*, *id.*, 8:53-67, 10:12-19, Figs. 3A, 5B. Thus, importing such a requirement into the claims is improper as a matter of law. *See Eko Brands, LLC v. Adrian Rivera Maynez Enter., Inc.*, 946 F.3d 1367, 1372-73 (Fed. Cir. 2020).

Google is also wrong to argue that "the specification confirms that 'naming' groups is different from the claimed 'zone scenes.'" G.Br., 24. To the contrary, the '885 Patent discloses that a name is one attribute of a "zone scene" that is established during creation of the "zone scene." '885 Pat., 8:52-61, 10:36-43, Figs. 5A, 7-8. This confirms that naming is a part of "zone scenes," and that a thematic name satisfies the "according to a common theme" aspect of Google's construction. Google's related suggestion that a zone scene's name cannot amount to a "common theme" because the '885 Patent discloses thematic names for "conventional" speaker groups (G.Br., 24) is based on a misreading of the patent. The disclosure Google points to describes the need in the prior art for a better way to group speakers so that a user can listen to audio on a first group in the "morning," a second group in the "evening," and a third group on the "weekend," where those groups have one or more overlapping speakers. *See* '885 Pat., 2:9-15. Nothing in that passage discloses if or how the prior art enabled such groups to be named. Regardless, whether the prior

art allowed a "conventional" group to be named is irrelevant to the question of whether a Google speaker group having a thematic name is "according to a common theme" – which it clearly is.

Google's other noninfringement arguments fail as well. Google contends that Sonos's interpretation of "common theme" is improper because it allegedly depends on what the user is "thinking" during creation of a "zone scene." G.Br., 23. This is not true. As explained, each Accused Google Player infringes under Google's proposed construction because each is capable of being included in previously saved groupings of players that are undisputedly "according to a common theme," such as a speaker group named "Morning" or "Garden." This infringement is not based on what the user is "thinking" – it is based on the functional capability of the Accused Google Players. No inquiry into the user's state of mind is needed.

Google also argues that it does not infringe because its speaker groups are "merely generic speaker groups, which were well known in speaker systems at the time." G.Br., 22. Of course, using "generic" technology is not an infringement defense, but regardless, Google is wrong – "zone scenes" such as Google's speaker groups are far from being "generic" or "well known."[10] Indeed, Google ignores the key aspects of a Google speaker group that make it a "zone scene" and distinguish it from a "generic speaker group" – namely, that it is a "***previously saved***" grouping of "zone players" that are "to be configured for synchronous playback of media ***when the [Google speaker group] is invoked***." '885 Pat., cl. 1; D.I. 126, App. A, 27. This allows an Accused Google Player to be included in a pre-saved speaker group that can be invoked at any future time, and also allows an Accused Google Player to be included in multiple different speaker groups—both of which are features that distinguish "zone scenes" such as Google's speaker groups from "generic speaker groups." *See, e.g.,* '885 Pat., 1:61-2:17, 8:42-45, 11:51-67; D.I. 209.03, ¶22-25.

IV. CONCLUSION

For the foregoing reasons, the Court should deny Google's motion for summary judgment.

[10] Nor would it show that '885 Patent claim 1 is invalid because there are other elements in the claim; a patent claim may have both old and new elements. *See, e.g.*, *Rosemount, Inc. v. Beckman Instruments, Inc.*, 727 F.2d 1540, 1546 (Fed. Cir. 1984) ("[A] combination may be patentable whether it be composed of elements all new, partly new, or all old").

Dated: May 5, 2022

By: */s/ Michael P. Boyea*

CLEMENT SETH ROBERTS
BAS DE BLANK
ALYSSA CARIDIS
EVAN D. BREWER

ORRICK, HERRINGTON & SUTCLIFFE LLP

SEAN M. SULLIVAN (*pro hac* vice)
MICHAEL P. BOYEA (*pro hac* vice)
COLE B. RICHTER (*pro hac* vice)

LEE SULLIVAN SHEA & SMITH LLP

Attorneys for Sonos, Inc.